ステップ30

Python [応用編]

ワークブック

CUTT
カットシステム

はじめに

本書は、Python で役に立つプログラムを作成したり、さまざまなデータをさまざまな形で利用したり操作する方法を習得することを目的とする書籍です。

本書では主に文字列や数値のデータを扱うことに焦点を当てています。

データは、テキストファイルやバイナリファイルとして保存される他に、CSV ファイル、Excel ファイル、データベースなどで保存されることがよくありますが、Python ではこれらのデータを比較的容易に扱うことができます。

また、こうしたデータの、平均値や標準偏差などの統計的な値を計算したり、データをグラフにして表示したりすることも Python ではとても容易です。

本書はこれらについて 30 ステップで学習します。

■ 対象読者

本書は、Python［基礎編］ワークブックに記載されているような Python の基本的な事柄をマスターした読者を対象としています。

■ 本書の構成

本書には以下の内容が含まれています。

- 乱数を使って一様分布や標準分布のデータを作成する方法
- コンソールからの基本的な入力方法
- 平均値や最大値／最小値、中央値、標準偏差などの 基本的な統計処理を python で行う方法
- データからグラフを描いて可視化する方法
- Python でテキストファイルやバイナリファイルへの入出力の方法
- Python で CSV ファイルや Excel ファイルを扱う方法
- Python でデータベースを扱うための基礎
- CSV ファイル、Excel ファイル、データベースファイルへの相互変換の方法

各ステップは次のような構成になっています。

要点

重要な事項の解説です。

ワーク

読者が取り組む問題です。ワークの解答は、カットシステムの Web ページからダウンロードできます。

■ 本書の表記

>>> Python のインタラクティブシェル（インタープリタ）のプロンプト（一次プロンプトともいう）を表します。Python のインタラクティブシェルでの実行例で >>> が掲載されていても、>>> は入力しません。

... Python のインタラクティブシェルの行の継続を表します。Python のインタラクティブシェルでの実行例で ... が掲載されていても、... は入力しません。

> OS のプロンプトを表します。Linux など UNIX 系の OS では、プロンプトは $、% などです。

リスト ex*m-n* 独立した Python のスクリプトファイルです。OS のコマンドライン（>、$、% など）から「python ex*m*-*n*.py」という形式で実行します（*m* と *n* には数字が入ります）。

■ ご注意

- 本書の内容は本書執筆時の状態で記述しています。Python のバージョンによっては本書の記述と実際とが異なる結果となる可能性があります。
- 本書のサンプルは、プログラミングを理解するために掲載するものです。実用的なプログラムとして提供するものではありませんので、ユーザーのエラーへの対処やセキュリティー、その他の面で省略してあるところがあります。

■ 本書に関するお問い合わせについて

本書に関するお問い合わせは、sales@cutt.co.jp にメールでお寄せください。なお、本書の記述から外れる内容についてのお問い合わせには対応できません。お問い合わせの際には下記事項を明記してくださいますようお願いいたします。

- 氏名
- 連絡先メールアドレス
- 書名
- 記載ページ
- お問い合わせ内容
- 実行環境

※ワークの解答は、以下の Web ページからダウンロードできます。

https://----------

目次

Step 01 乱数 7

1.1 乱数の生成……7　1.2 リストへの複数のランダム値の保存……7
1.3 整数の乱数……8　1.4 正規分布の乱数……9

Step 02 値の入出力 10

2.1 文字列の入力……10　2.2 リストへの複数の値の保存……10
2.3 文字列から数値への変換……11　2.4 タプルの作成……12

Step 03 データのグラフ化1 14

3.1 折れ線グラフ……14

Step 04 データのグラフ化2 18

4.1 円グラフ……18

Step 05 データのグラフ化3 22

5.1 ヒストグラム……22　5.2 散布図……24

Step 06 関数のグラフ 26

6.1 一次関数のグラフ……26　6.2 二次関数のグラフ……27
6.3 三角関数のグラフ……28

Step 07 統計値を求める1 30

7.1 平均値……30　7.2 加重平均……31　7.3 最大値と最小値……31
7.4 中央値……32　7.5 最頻値……32

Step 08 統計値を求める2 34

8.1 分散……34　8.2 標準偏差……34

Step 09 テキストファイルの読み書き 36

9.1 ファイルへのテキストの出力……36　9.2 ファイルへの追加……37
9.3 ファイルからの読み込み……38　9.4 複数行の読み込み……38

Step 10 テキストデータから統計値を求める 40

10.1 代表値……40　10.2 散布度……42

Step 11 テキストデータからグラフを作成する 44

11.1 折れ線グラフ……44　11.2 ヒストグラム……46

Step 12 コマンドライン引数.....48

12.1 コマンドライン……48　12.2 コマンドライン引数……48
12.3 main()……49

Step 13 バイナリファイルの読み書き.....52

13.1 バイナリファイル……52　13.2 ファイルへのバイナリ出力……53
13.3 ファイルからバイナリ入力……53　13.4 ファイルのコピー……54

Step 14 バイナリデータから統計値を求める.....56

14.1 代表値……56　14.2 散布度……57

Step 15 バイナリデータの16進ダンプ.....58

15.1 ファイルの16進表示……58　15.2 ファイルの16進ASCII表示……60

Step 16 CSVファイルの読み書き.....62

16.1 CSVファイル……62　16.2 CSVファイルへの出力……62
16.3 CSVファイルからの入力……64　16.4 CSVのそれぞれの値の取得……65

Step 17 CSVデータから統計値を求める.....66

17.1 代表値……66　17.2 散布度……67

Step 18 CSVデータからグラフを作成する.....68

18.1 ヒストグラム……68　18.2 散布図……69

Step 19 Excelファイルの読み書き.....70

19.1 Excelについて……70　19.2 Excelファイルからの読み込み……70
19.3 Excelファイルへの書き込み……72

Step 20 Excelデータから統計値を求める.....74

20.1 代表値……74　20.2 散布度……74

Step 21 Excelデータからグラフを作成する.....76

21.1 ヒストグラム……76　21.2 散布図……77

Step 22 データベースの基礎.....78

22.1 データベース……78　22.2 データベースの構造……78
22.3 カレントレコードとキー……79　22.4 SQL……79
22.5 データベースの操作方法……81

Step 23 DBの作成とデータ登録.....82

23.1 データベースの作成……82　23.2 テーブルの作成……82
23.3 レコードの登録……83

Step 24 DBのデータの検索.....86

24.1 すべてのレコードの取得……86　24.2 データの検索……87

Step 25 DBのデータ操作.....89

25.1 レコードの登録……89　25.2 複数のレコードの登録……90
25.3 レコードの削除……91　25.4 レコードの更新と挿入……91

Step 26 DBのデータから統計値を求める.....93

26.1 代表値……93　26.2 散布度……94

Step 27 DBのデータからグラフを作成する.....96

27.1 ヒストグラム……96　27.2 散布図……96

Step 28 CSVデータの変換.....98

28.1 Excelファイルへの変換……98
28.2 データベースファイルへの変換……99

Step 29 Excelデータの変換.....102

29.1 CSVファイルへの変換……102
29.2 データベースファイルへの変換……103

Step 30 データベースの変換.....106

30.1 CSVファイルへの変換……106
30.2 Excelファイルへの変換……107

付録A モジュールのインストール.....109

A.1 モジュールのインストール……109
A.2 モジュールのアップグレード……109

付録B トラブル対策.....110

B.1 Pythonの起動……110　B.2 Python実行時のトラブル……110
B.3 データベースのトラブル……113

索 引.....115

Step 01

乱数

要点

1.1 乱数の生成

さまざまな場面でランダムな値を必要とすることがあります。random というモジュールを使うと、ランダムな値（乱数）を生成することができます。

random モジュールを使う前に import 文でこのモジュールをインポートします（random モジュールは標準モジュールなのでインストールしなくても使うことができます）。

```
>>> import random
```

random.random() は 0.0 以上 1.0 未満の乱数を生成します。生成される値はコードを実行するごとに異なります。

```
>>> random.random()
0.8543558099577775
>>> random.random()
0.34298714667620445
```

1.2 リストへの複数のランダム値の保存

入力された複数の値をリストを使って保存することもできます。次の例は for 文を使う例です。

```
data = []                # 空のリストを作る
for i in range(5):       # 5回繰り返す
    val = random.random()
    data.append(val)
```

上のプログラムは、はじめに空のリスト data を宣言しておき、乱数を生成しては data に保存するということを for 文で 5 回繰り返します。

次の例は、5 個の乱数を生成してリストに保存し、保存された値を出力するスクリプトの例です。

リスト● ex1-1.py

```
import random
data = []                # 空のリストを作る
for i in range(5):       # 5回繰り返す
    val = random.random()
    data.append(val)

print('生成された値')
for val in data:         # 保存された値を出力する
    print(val)
```

スクリプトを実行するには、python に続けてスクリプトファイルの名前を指定します。たとえば、Windows でスクリプトを C:\PythonWork2\step01 に保存した場合には次のようにします。

```
C:\PythonWork2\step01>python ex1-1.py
```

実行例を次に示します（実際に生成される値はプログラムを実行するごとに変わります）。

```
生成された値
0.26702902538545925
0.40090688194616275
0.03069287734585746
0.2465440685506376
0.47384060147024876
```

1.3 整数の乱数

整数の乱数は random.randint() で生成することができます。

random.randint() の書式は次のとおりです。

```
random.randint(a, b)
```

a は生成する整数の最小値、b は最大値です。つまり、random.randint(a, b) は a 以上 b 以下の整数を生成します。たとえば、1 以上 10 以下のランダムな整数値を生成したいときには、次のようにします。

```
>>> import random
>>> random.randint(1, 10)
7
>>> random.randint(1, 10)
10
```

1.4 正規分布の乱数

これまでに示したrandomモジュールを使って生成する乱数は、どの値の確率も等しい一様分布の乱数でした。しかし、世の中の値の中には、正規分布に従うものが多数あります。正規分布は平均値付近に多くの値があり、そこから離れるに従って数が減る分布です。

Numpyというモジュールを使うと、正規分布に従うランダムな値（乱数）を生成することができます。

NumpyはPythonの外部モジュールなので、まだインストールされていない場合は、次のコマンドをOSのプロンプトで実行してインストールする必要があります（Numpyは、モジュール名としては先頭文字が小文字のnumpyを使います）。

```
>python -m pip install numpy
```

numpyモジュールを使う前にこのモジュールをインポートしてnpという名前を付けます。

```
import numpy as np
```

np.random.normal()は次の書式で平均（loc）がl、標準偏差（scale）がs、サイズ（size）がnの乱数の配列を生成します。

```
np.random.normal(loc=l, scale=s, size=n)
```

たとえば、次の例は平均が5.0、標準偏差が2の乱数が5個入っている配列を生成します。二次元の配列を生成するには、sizeにタプルで配列の要素数を指定します。

```
>>> np.random.normal(loc=5.0, scale=2, size=5)
array([7.16076578, 5.16835757, 1.47696127, 5.17493545, 4.55438336])
>>> np.random.normal(loc=5.0, scale=2, size=(3, 3))
array([[8.4592616 , 3.4221892 , 2.97035133],
       [6.85624644, 6.60802539, 9.25685457],
       [6.59803827, 7.23600677, 6.87387991]])
```

ワーク

1-1

1から100の範囲の整数の乱数を10個生成し、その値を出力するプログラムを作りましょう。

1-2

平均が10.0、標準偏差が3.0の乱数が10個入っている配列を生成し、その値を出力するプログラムを作りましょう。

Step 02

値の入出力

要点

2.1 文字列の入力

キーボードからの値の入力には、input() を使うことができます。input() の引数には、入力を促すためのプロンプトを指定することができます。

たとえば、「名前 :」と表示して入力を促し、入力された値を name という変数に保存し、変数の内容を表示するときには次のようにします。

```
>>> name = input('名前:')
名前:山田花子
>>> name
'山田花子'
```

2.2 リストへの複数の値の保存

入力された複数の値をリストを使って保存することもできます。

```
names = []              # 空のリストを作る

for i in range(5):      # 名前を入力することを5回繰り返す
    name = input('名前:')
    names.append(name)
```

次の例は、「quit」という文字列が入力されるまで複数の名前を入力してリストに保存し、保存された名前を出力するスクリプトの例です。

リスト● ex2-1.py

```
names = []
while 1:
    name = input('名前:')
    if name == 'quit' :
        break
    names.append(name)

print('入力された名前')
```

```
for name in names:
    print(name)
```

上のプログラムでは、はじめに空のリスト names を宣言しておきます。そして、文字列「quit」が入力されるまで while 文で繰り返し名前を入力して、入力された名前を append() でリストに追加します。最後に、for 文を使って入力された一連の名前を出力します。

実行例を次に示します。

```
名前:山田花子
名前:Jimmy
名前:斎藤一郎
名前:quit
入力された名前
山田花子
Jimmy
斎藤一郎
```

2.3 文字列から数値への変換

input() で入力すると、たとえ数値を入力しても文字列として保存されます。

```
>>> score = input('成績:')
成績:88
>>> score
'88'
>>> type(score)
<class 'str'>
```

文字列を整数に変換するには、int() を使います。

```
>>> score = int(score) # 数値に変換
>>> score
88
>>> type(score)
<class 'int'>
```

つまり次の 2 行で整数値の入力を行うことができます。

```
score = input('成績:')
score = int(score)      # 数値に変換
```

なお、数値以外の値を int() で変換しようとすると、エラーになります。また、実数値に変換する場合には float() を使います。

2.4 タプルの作成

複数の値を入力して保存したいときには、値を（ ）で囲んでタプル（Tuple）にして保存することができます。

```
    name = input('名前:')
    eng = input('英語:')
    math = input('数学:')
    eng = int(eng)
    math = int(math)
    data = (name, eng, math)
```

次の例は、「quit」という文字列が入力されるまで複数の名前と英語と数学の成績を入力してリスト data に保存し、保存されたデータを出力するスクリプトの例です。

リスト● ex2-2.py

```
data = []

while 1:
    name = input('名前:')
    if name == 'quit' :
        break
    eng = input('英語:')
    math = input('数学:')
    eng = int(eng)
    math = int(math)
    data.append( (name, eng, math) )

print('名前\t\t英語\t数学')
for name, eng, math in data:
    print(f"{name:16s}{eng:3d}{math:8d}")
```

上のリストの「print('名前\t\t英語\t数学')」の中の \t は水平タブを表す制御文字で、出力を 8 文字ごとに位置を揃えるために使っています。

「print(f"{name:16s}{eng:3d}{math:8d}")」の中の f で始まる文字列は出力する書式を設定しています。16s は 16 文字の文字列を表し（日本語には対応していません）、3d は 3 桁の数値を表します。

実行例を次に示します。

```
名前:Jimmy
英語:88
数学:76
名前:Kelly
英語:100
数学:79
```